Проект по Разпределени софтуерни архитектури

**Изобразяване на фрактал**

Изготвил: Виктор Христов, СИ 3, №62151

Дата: 13.6.2020

Съдържание

[1. Увод 2](#_Toc42947320)

[1.1. Цел и предназначение на проектираното приложение 2](#_Toc42947321)

[1.2. Анализ на решения на задачата на Манделброт чрез имплементация на паралелни алгоритми 2](#_Toc42947322)

[1.3. Нефункционален анализ 3](#_Toc42947323)

[2. Проектиране 3](#_Toc42947324)

[2.1. Функционално проектиране 3](#_Toc42947325)

[2.2. Нефункционално проектиране 3](#_Toc42947326)

[3. Тестване 3](#_Toc42947327)

[4. Списък източници 3](#_Toc42947328)

# Увод

# Цел и предназначение на проектираното приложение

Целта на този проект е да се създаде програма, която генерира изображение на множество на Манделброт с функция **Zn+1=C∗cos(Zn)** за комплексно число C чрез реализиране на паралелен алгоритъм.

Множеството на Манделброт е вид фрактал. Това са фигури, които рекурсивно съдържат себеподобни фигури по своите граници. Всеки фрактал се изобразява чрез множество от комплекси числа. Комплексните числа се пресмятат чрез използване на комплексна равнина. Това е координатна система, при която реалната част на числото се представя чрез абцисната ос, а имагинерната - чрез ординатната. След като се знае стойността на комплексното число, то тогава може да се определи дали то принадлежи към конкретния фрактал. Прилага се следната система:

1. Z0=0
2. Zn+1=C∗cos(Zn)

Ако след избран от програмиста брой итерации на функцията не може да се определи дали комплексното число клони към безкрайност, то тогава се приема, че принадлежи към това множество на Манделброт.

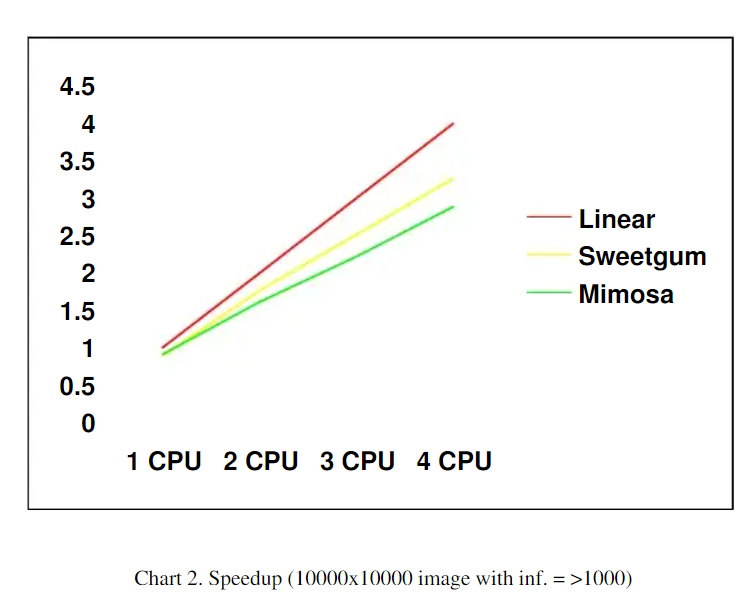
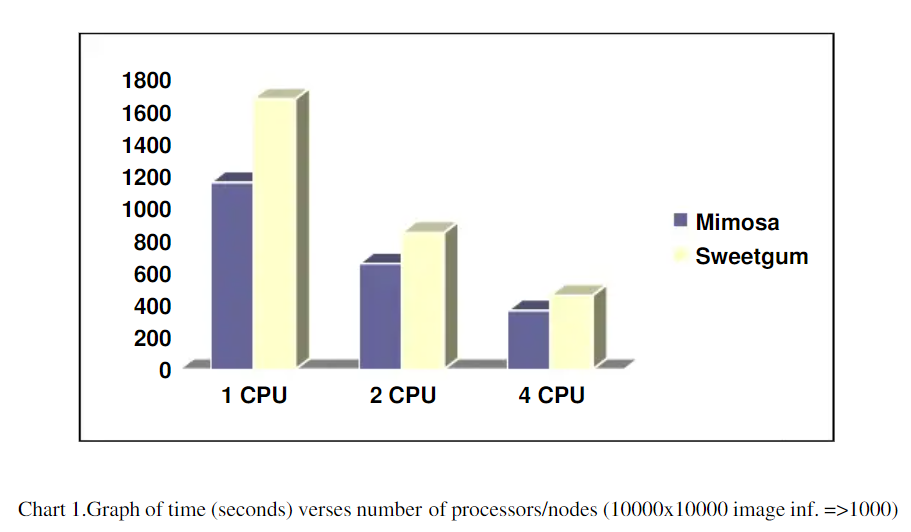
За да се генерира пълното изображение, ще е нужно да се изчисли всяка точка от зададената комплексна равнина, независимо дали е част от множеството на Манделброт или не. Единствено след като е приключило изчислението ще може да се запази генерираното изображение във визуален формат.

# Анализ на решения на задачата на Манделброт чрез имплементация на паралелни алгоритми

В тази подточка ще се разгледат три решения на задачата за изобразяване на множество на Манделброт, както и една допълнителна статия. Всяко от трите решения обяснява какво представлява задачата, как чрез комплекси числа се изобразява на координатна система и как се пресмята дали едно число е част от множеството. Поради това при този анализ ще се обърне внимание само на по-значителните части от статиите, главно какво алгоритми прилагат решенията.

1. Parallel Implementation and Analysis of Mandelbrot Set Construction [[1]](#_Списък_източници)

* Авторът обяснява „escape time“ алгоритъма, чрез който се извършва оцветяването на всеки пиксел от изображението. Алгоритъмът се състои от многократно изчисляване на функцията Zn+1=Zn2 + C. Константата C се определя спрямо координатите на пиксела, който в момента се пресмята, а броят максимални итерации е предварително зададен от програмиста. При всяка итерация се извършва проверка дали е удовлетворено условието за излизане.
  + Ако е изпълнено това условие, то се приема, че пиксела е извън множеството на Манделброт и се оцветява спрямо броя на итерации, които се били нужни, за да се изпълни условието.
  + Ако всички итерации приключат без да се изпълни условието, то се приема, че пиксела принадлежи на множеството и се оцветява в един конкретен цвят, като общоприетия вариант е черен цвят.
* Авторът обръща внимание на проблема, че при разделянето на изображението на големи задания се получа лошо балансиране на работата на процесите, и предлага своето решение.
* Идеята на неговия паралелен алгоритъм е всеки процес да изчислява случайно избрани пиксели от изображението. Инициализират се два вектора, които представляват интервала на координатите X и Y от комплексната равнина. На случаен принцип се генерира число, спрямо което стойностите на векторите се разбъркват. След това векторите се използват, за да се генерира изображението.
* Комуникацията между процесите е минимизирана. Преди да се извършат изчисленията, процесите получават информация за векторите, след което използват тази информация, за да пресметнат заданията, които са им разпределени. Накрая главния процес събира всичките задания и сглобява пълното изображение.



* Авторът е показал следните графики като резултат от проведените тестове. Полученото ускорение е близо до линейното, но не може да се направи точен извод, защото е тествано единствено до 4 процеса. Още от изпълнение с 3 процеса започва да се спада ефективността, което подсказва че при тестване с повече от 4 процеса резултатите няма да са задоволителни.
* Под въпрос е колко е правилно да се използва случайно-генерирано число при разбъркване на стойностите на векторите. Случайността призовава единствено допълнителен хаос и несигурност в програмата. В повечето случаи ще се получи разбъркване на координатите по такъв начин, че всеки процес да обработва равномерно-балансирани задания. Но в някои случаи разбъркванията ще доведат до концентриране на точките от множеството в малък брой задания и няма да реши проблема на едрата грануларност. Този проблем щеше да си проличи, ако авторът беше показал подробно тестване на алгоритъма.
* Авторът описва своя алгоритъм на високо ниво и без показване на примерен код, поради което възникват въпроси относно реализацията на паралелния алгоритъм. Неговият подход решава проблема за балансирането на работата в повечето случаи, но не се обяснява как точно заданията се разпределят между процесите след като те получат информацията от векторите.

1. Parallel Fractal Image Generation - A Study of Generating Sequential Data with Parallel Algorithms [[2]](#_Списък_източници)

* Авторът на тази статия споменава важността от броя на итерациите при изчисляване дали някоя точка принадлежи на множеството на Манделброт. Някои точки се нуждаят от огромен брой итерации, за да определи дали числото клони към безкрайност.
* Колкото повече итерации се извършват, толкова по-правилно ще е крайното изображение. Но от друга гледна точка, колкото повече итерации се извършват, толкова по-бавно ще се генерира изображението. Поради това е важно да има компромис между точността и бързодействието на алгоритъма.

1. Parallel Mandelbrot in Julia, C++, and OpenCL [[3]](#_Списък_източници)
2. Parallel Processing and The Mandelbrot Set [[4]](#_Списък_източници)

# Нефункционален анализ

1. Технологии
2. Модел на обслужването
3. Софтуерен модел
4. Коментар и обосновка на избраното решение

# Проектиране

# Функционално проектиране

# Нефункционално проектиране

# Тестване

# Списък източници

[1] Isaac K. Gäng, David Dobson, Jean Gourd and Dia Ali,

Parallel Implementation and Analysis of Mandelbrot Set Construction, <https://www.academia.edu/1399383/Parallel_Implementation_and_Analysis_of_Mandelbrot_Set_Construction>

[2] Matthias Book,

Parallel Fractal Image Generation - A Study of Generating Sequential Data with Parallel Algorithms,

presented May 3, 2001,

<http://matthiasbook.de/papers/parallelfractals/index.html>

[3] Distrust Simplicity,

Parallel Mandelbrot in Julia, C++, and OpenCL, January 9, 2015,

<http://distrustsimplicity.net/articles/mandelbrot-speed-comparison/>

[4] Carmen Pughineanu,

Parallel Processing and The Mandelbrot Set,

<http://www.afahc.ro/ro/revista/Nr_2_2008/ART_CARMEN.pdf>